A Fast Algorithm for Computing the p-Curvature

Alin Bostan, Xavier Caruso, Éric Schost

A Fast Algorithm for Computing the p-Curvature

Alin Bostan
Inria (France)
alin.bostan@inria.fr

Xavier Caruso
Université Rennes 1
xavier.caruso@normalesup.org

Éric Schost
Western University
eschost@uwo.ca

ABSTRACT

We design an algorithm for computing the p-curvature of a differential system in positive characteristic p. For a system of dimension r with coefficients of degree at most d, its complexity is $O^{\sim}(pdr^{\omega})$ operations in the ground field (where ω denotes the exponent of matrix multiplication), whereas the size of the output is about pdr^2. Our algorithm is then quasi-optimal assuming that matrix multiplication is (*i.e.* $\omega = 2$). The main theoretical input we are using is the existence of a well-suited ring of series with divided powers for which an analogue of the Cauchy–Lipschitz Theorem holds.

Categories and Subject Descriptors:
I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation – *Algebraic Algorithms*

Keywords: Algorithms, complexity, differential equations, p-curvature.

1. INTRODUCTION

We study in this article algorithmic questions related to linear differential systems in positive characteristic. Let k be an arbitrary field of prime characteristic p, and A be an $r \times r$ matrix with entries in the field $k(x)$ of rational functions over k. A simple-to-define, yet very important object attached to the differential system $Y' = AY$ is its so-called *p-curvature*. It is the p-th iterate ∂_A^p of the map $\partial_A : k(x)^r \to k(x)^r$ that sends v to $v' - Av$. It turns out that it is $k(x)$-linear. It is moreover classical that its matrix with respect to the canonical basis of $k(x)^r$ is equal to the term A_p of the recursive sequence $(A_i)_i$ defined by

$$A_1 = -A \quad \text{and} \quad A_{i+1} = A_i' - A \cdot A_i \quad \text{for} \quad i \geq 1. \qquad (1)$$

In all what follows, we will thus deliberately identify the matrix A_p with the *p-curvature of $Y' = AY$*. The above recurrence yields an algorithm for computing it, sometimes referred to as *Katz's algorithm*.

The p-curvature is related to solutions; it measures to what extent the usual Cauchy–Lipschitz theorem applies in characteristic p. More precisely, at an ordinary point, the system $Y' = AY$ admits a fundamental matrix of power series solutions in $k[[x]]$ if and only if the p-curvature A_p vanishes. In this case, the system $Y' = AY$ even admits a fundamental matrix of solutions which are rational functions in $k(x)$. More generally, the dimension of the kernel of A_p is equal to the dimension of the space of rational function solutions of $Y' = AY$.

The primary importance of the notion of p-curvature relies in its occurrence in one of the versions of the celebrated Grothendieck–Katz conjecture [19, 20, 12, 30]. This conjecture, first formulated by Alexandre Grothendieck in the late 1960s, is a local-global principle for linear differential systems, which states that a linear differential system with rational function coefficients over a function field admits a fundamental matrix of algebraic solutions if and only if its p-curvatures vanish for almost all primes p.

In computer algebra, p-curvature has been introduced by van der Put [22, 23], who popularized it as a tool for factoring differential operators in characteristic p. Cluzeau [13] generalized the approach to the decomposition of differential systems over $k(x)$. The p-curvature has also been used by Cluzeau and van Hoeij [14] as an algorithmic filter for computing exponential solutions of differential operators in characteristic zero.

Computing efficiently the p-curvature is in itself a challenging problem, especially for large values of p. Our initial motivation for studying this question emerged from concrete applications, in lattice path combinatorics [6, 7] and in statistical physics [3]. In this article, we address the question of the computation of A_p in good complexity, with respect to three parameters: the dimension r of the system $Y' = AY$, the maximum degree d of the rational function entries of A, and the characteristic p of the ground field. In terms of these quantities, the arithmetical size of A_p is generically proportional to pdr^2 if $r > 1$.

Previous work. Cluzeau [13, Prop. 3.2] observed that the direct algorithm based on recurrence (1) has complexity $O^{\sim}(p^2dr^{\omega})$, where ω is the matrix multiplication exponent and the soft-O notation $O^{\sim}(\,)$ hides polylogarithmic factors. Compared to the size of the p-curvature, this cost is good with respect to r and d, but not to p. The first subquadratic algorithm in p, of complexity $O^{\sim}(p^{1+\omega/3})$, was designed in [9, §6.3]. In some special cases, additional partial results were obtained in [9], notably an algorithm of quasi-linear cost $O^{\sim}(p)$ for certain systems of order $r = 2$. However, the question of designing a general algorithm for computing A_p with quasi-linear complexity in p remained open. In a related, but

ISSAC'15, July 6–9, 2015, Bath, United Kingdom..

ACM 978-1-4503-3435-8/15/07 ...$15.00.
http://dx.doi.org/10.1145/2755996.2756674.

different direction, the article [4] proposed an algorithm for computing the characteristic polynomial of the p-curvature in time essentially linear in $\sqrt{p}$, without computing A_p itself.

Contribution. We prove that the p-curvature A_p can be computed in quasi-linear time with respect to p. More precisely, our main result (Theorem 4.2) states that $O^{\sim}(pdr^{\omega})$ operations in k are sufficient for this task. This complexity result is quasi-optimal not only with respect to the main parameter p, but also to d; with respect to the dimension r, it is as optimal as matrix multiplication. Moreover the algorithm we obtain is highly parallelizable by design. The key tools underlying the proof of Theorem 4.2 are the notion of divided power rings in characteristic p, and a new formula for the p-curvature (Propositions 4.3 and 4.4) in terms of divided power series. Crucial ingredients are the fact that a Cauchy–Lipschitz theorem for differential systems holds over divided power rings (Proposition 3.4) and the fact that Newton iteration can be used to efficiently compute (truncations of) fundamental matrices of divided power solutions.

Structure of the paper. In Section 2, we recall the main theoretical properties of the basic objects used in this article. Section 3 is devoted to the existence and the computation of solutions of differential systems in divided power rings. In Section 4, we move to the main objective of the article, the computation of the p-curvature: after relating A_p to the framework of divided powers, we describe our main algorithm for A_p, of complexity $O^{\sim}(pdr^{\omega})$. We conclude in Section 5 by describing the implementation of our algorithm and some benchmarks.

Complexity measures. Throughout this article, we estimate the cost of our algorithms by counting arithmetic operations in the base ring or field at unit cost.

We use standard complexity notations. The letter ω refers to a feasible exponent for matrix multiplication (*i.e.* there exists an algorithm for multiplying $n \times n$ matrices over a ring $\mathfrak{A}$ with at most $O(n^{\omega})$ operations in $\mathfrak{A}$); the best known bound is $\omega < 2.3729$ from [15]. The soft-O notation $O^{\sim}(\cdot)$ indicates that polylogarithmic factors are omitted; in particular, we will use the fact that many arithmetic operations on univariate polynomials of degree d can be done in $O^{\sim}(d)$ operations: addition, multiplication, Chinese remaindering, *etc*, the key to these results being fast polynomial multiplication [27, 26, 11, 18]. A general reference for these questions is [17].

2. THEORETICAL SETTING

We introduce and briefly recall the main properties of the theoretical objects we are going to use in this article. All the material presented in this section is classical; a general reference is [24].

Definitions and notations. Let $\mathfrak{A}$ be a commutative ring with unit. We recall that a *derivation* on $\mathfrak{A}$ is an additive map $' : \mathfrak{A} \to \mathfrak{A}$, satisfying the Leibniz rule $(fg)' = f'g + fg'$ for all $f, g \in \mathfrak{A}$. The image f' of f under the derivation is called the *derivative* of f. From now on, we assume that $\mathfrak{A}$ is equipped with a derivation. A *differential system* with coefficients in $\mathfrak{A}$ is an equation of the form $Y' = AY$ where A is a given $r \times r$ matrix with coefficients in $\mathfrak{A}$ (for a certain positive integer r), the unknown Y is a column vector of length r and Y' denotes the vector obtained from Y by taking the derivative component-wise. The integer r is called the *dimension* of the system. We recall briefly that a linear differential equation:

$$a_r y^{(r)} + \cdots + a_1 y' + a_0 y = 0 \quad (\text{with } a_i \in \mathfrak{A}) \tag{2}$$

can be viewed as a particular case of a differential system. Indeed, defining the companion matrix

$$C = \begin{pmatrix} & & -\frac{a_0}{a_r} \\ 1 & & -\frac{a_1}{a_r} \\ & \ddots & \vdots \\ & 1 & -\frac{a_{r-1}}{a_r} \end{pmatrix} \tag{3}$$

and $A = {}^{t}C$, the solutions of the system $Y' = AY$ are exactly the vectors of the form ${}^{t}(y, y', \ldots, y^{(r-1)})$ where y is a solution of (2). In this correspondence, the order of the differential equation agrees with the dimension of the associated differential system.

Differential modules. A *differential module* over $\mathfrak{A}$ is a pair (M, ∂) where M is an $\mathfrak{A}$-module and $\partial : M \to M$ is an additive map satisfying a Leibniz-like rule, which is:

$$\forall f \in \mathfrak{A}, \forall x \in M, \quad \partial(fx) = f' \cdot x + f \cdot \partial(x). \tag{4}$$

There exists a canonical one-to-one correspondence between differential systems and differential modules (M, ∂) for which $M = \mathfrak{A}^r$ for some r: to a differential system $Y' = AY$ of dimension r, we attach the differential module $(\mathfrak{A}^r, \partial_A)$ where $\partial_A : \mathfrak{A}^r \to \mathfrak{A}^r$ is the function mapping X to $X' - AX$. Under this correspondence, the solutions of $Y' = AY$ are exactly vectors in the kernel of ∂_A.

To a differential equation as (2), one can associate the *differential operator* $L = a_r\partial^r + a_{r-1}\partial^{r-1} + \cdots + a_1\partial + a_0$; it lies in the non-commutative ring $\mathfrak{A}\langle\partial\rangle$, endowed with the usual addition of polynomials and a multiplication ruled by the relation $\partial \cdot f = f \cdot \partial + f'$ for all $f \in \mathfrak{A}$ (note that, as often in the literature, we are using ∂ to denote either the structure map of a differential module, and a non-commutative indeterminate).

Then, if a_r is a unit in $\mathfrak{A}$, one can further associate to L the quotient $\mathfrak{A}\langle\partial\rangle/\mathfrak{A}\langle\partial\rangle L \simeq \mathfrak{A}^r$. The differential structure inherited from $\mathfrak{A}\langle\partial\rangle$ makes it a differential module with structure map $X \in \mathfrak{A}^r \mapsto X' + CX$, where C is the companion matrix defined above; in other words, this is the module $(\mathfrak{A}\langle\partial\rangle/\mathfrak{A}\langle\partial\rangle L, \partial_{-C})$, with the previous notation.

Scalar extension. Let $\mathfrak{A}$ and $\mathfrak{B}$ be two rings equipped with derivations and let $\varphi : \mathfrak{A} \to \mathfrak{B}$ be a ring homomorphism commuting with derivation. From a given differential system $Y' = AY$ with coefficients in $\mathfrak{A}$, one can build a differential system over $\mathfrak{B}$ by applying φ: it is $Y' = \varphi(A)Y$, where $\varphi(A)$ is the matrix obtained from A by applying φ entry-wise.

This operation admits an analogue at the level of differential modules: to a differential module (M, ∂) over $\mathfrak{A}$, we attach the differential module $(M_{\mathfrak{B}}, \partial_{\mathfrak{B}})$ over $\mathfrak{B}$ where $M_{\mathfrak{B}} = \mathfrak{B} \otimes_{\varphi,\mathfrak{A}} M$ and $\partial_{\mathfrak{B}} : M_{\mathfrak{B}} \to M_{\mathfrak{B}}$ is defined by:

$$\forall f \in \mathfrak{B}, \forall x \in M, \quad \partial_{\mathfrak{B}}(f \otimes x) = f' \otimes x + f \otimes \partial(x).$$

It is easily seen that if (M, ∂) is a differential module associated to the system $Y' = AY$ then $(M_{\mathfrak{B}}, \partial_{\mathfrak{B}})$ is that associated to the system $Y' = \varphi(A)Y$.

The p-curvature. Let k be any field of characteristic p. We assume here that $\mathfrak{A}$ is the field $k(x)$ — consisting of rational functions over k — equipped with the standard derivation.

The *p-curvature* of a differential module (M, ∂) over $k(x)$ is defined as the mapping $\partial^p : M \to M$. It follows from the Leibniz rule (4) and the fact that the p-th derivative of any element of $k(x)$ vanishes that the p-curvature is $k(x)$-linear.

This definition extends to differential systems as follows: the p-curvature of the system $Y' = AY$ is the $k(x)$-linear map $\partial_A^p : M_A \to M_A$ where (M_A, ∂_A) is the corresponding differential module. One can check that the matrix of ∂_A^p (in the canonical basis of M_A) is the p-th term of the recursive sequence (A_i) defined in (1).

Considering again a differential operator L and the associated differential module $(\mathfrak{A}\langle\partial\rangle/\mathfrak{A}\langle\partial\rangle L, \partial_{-C})$, for the associated companion matrix C, we obtain the usual recurrence $A_1 = C$ and $A_{i+1} = A_i' + C \cdot A_i$. The p-curvature of $\mathfrak{A}\langle\partial\rangle/\mathfrak{A}\langle\partial\rangle L$ will simply be called the p-curvature of L.

3. SERIES WITH DIVIDED POWERS

In all this section, we let ℓ be a ring in which p vanishes. We recall the definition of the divided power ring over ℓ, and its main properties — mainly, a Cauchy–Lipschitz theorem that will allow us to compute solutions of differential systems. We show how some approaches that are well-known for power series solutions carry over without significant changes in this context. Most results in this section are not new; those from §3.1 and §3.2 are implicitly contained in [1, 2], while the theoretical basis of §3.3 is similar to [21].

3.1 The ring $\ell[[t]]^{\mathrm{dp}}$

Let $\ell[[t]]^{\mathrm{dp}}$ be the ring of formal series of the form:

$$f = a_0 + a_1\gamma_1(t) + a_2\gamma_2(t) + \cdots + a_i\gamma_i(t) + \cdots \qquad (5)$$

where the a_i's are elements of ℓ and each $\gamma_i(t)$ is a symbol which should be thought of as $\frac{t^i}{i!}$. The multiplication on $\ell[[t]]^{\mathrm{dp}}$ is defined by the rule $\gamma_i(t) \cdot \gamma_j(t) = \binom{i+j}{i} \cdot \gamma_{i+j}(t)$.

Remark 3.1. *The ring $\ell[[t]]^{\mathrm{dp}}$ is not the PD-envelope in the sense of [1, 2] of $\ell[[t]]$ with respect to the ideal (t) but its completion for the topology defined by the divided powers ideals. Taking the completion is essential to have an analogue of the Cauchy–Lipschitz Theorem (*cf *Proposition 3.4).*

Invertible elements of $\ell[[t]]^{\mathrm{dp}}$ are easily described: they are exactly those for which the "constant" coefficient a_0 is invertible in ℓ. The ring $\ell[[t]]^{\mathrm{dp}}$ is moreover endowed with a derivation defined by $f' = \sum_{i=0}^{\infty} a_{i+1}\gamma_i(t)$ for $f = \sum_{i=0}^{\infty} a_i\gamma_i(t)$. It then makes sense to consider differential systems over $\ell[[t]]^{\mathrm{dp}}$. A significant difference with power series is the existence of an integral operator: it maps f as above to $\int f = \sum_{i=0}^{\infty} a_i\gamma_{i+1}(t)$ and satisfies $(\int f)' = f$ for all f.

Divided power ideals. For all positive integers N, we denote by $\ell[[t]]^{\mathrm{dp}}_{\geq N}$ the ideal of $\ell[[t]]^{\mathrm{dp}}$ consisting of series of the form $\sum_{i\geq N} a_i\gamma_i(t)$. The quotient $\ell[[t]]^{\mathrm{dp}}/\ell[[t]]^{\mathrm{dp}}_{\geq N}$ is a free ℓ-module of rank N and a basis of it is $(1, \gamma_1(t), \ldots, \gamma_{N-1}(t))$. In particular, for $N = 1$, the quotient $\ell[[t]]^{\mathrm{dp}}/\ell[[t]]^{\mathrm{dp}}_{\geq 1}$ is isomorphic to ℓ: in the sequel, we shall denote by $f(0) \in \ell$ the reduction of an element $f \in \ell[[t]]^{\mathrm{dp}}$ modulo $\ell[[t]]^{\mathrm{dp}}_{\geq 1}$. On the writing (5), it is nothing but the constant coefficient a_0 in the expansion of f.

We draw the reader's attention to the fact that $\ell[[t]]^{\mathrm{dp}}_{\geq N}$ is *not* stable under derivation, so the quotients $\ell[[t]]^{\mathrm{dp}}/\ell[[t]]^{\mathrm{dp}}_{\geq N}$ do *not* inherit a derivation.

Relationship with $\ell[t]$. There exists a natural map $\varepsilon : \ell[t] \to \ell[[t]]^{\mathrm{dp}}$ taking a polynomial $\sum_i a_i t^i$ to $\sum_{i=0}^{p-1} a_i i! \cdot \gamma_i(t)$. The latter sum stops at $i = p-1$ because $i!$ becomes divisible by p after that. Clearly, the kernel of ε is the principal ideal generated by t^p. Hence ε factors through $\ell[t]/t^p$ as follows:

$$\ell[t] \xrightarrow{\mathrm{pr}} \ell[t]/t^p \xrightarrow{\iota} \ell[[t]]^{\mathrm{dp}} \qquad (6)$$

where pr is the canonical projection taking a polynomial to its reduction modulo t^p. We observe moreover that the ideal $t^p\ell[t]$ is stable under derivation and, consequently, that the quotient ring $\ell[t]/t^p$ inherits a derivation. Furthermore, the two mappings in (6) commute with the derivation.

3.2 Computations with divided powers

It turns out that the $\gamma_n(t)$'s can all be expressed in terms of only few of them, resulting in a more flexible description of the ring $\ell[[t]]^{\mathrm{dp}}$. To make this precise, we set $t_i = \gamma_{p^i}(t)$ and first observe that $t_i^n = n! \cdot \gamma_{np^i}(t)$ for all i and n; this is proved by induction on n, using the equalities

$$t_i^{n+1} = n! \cdot \gamma_{np^i}(t) \cdot \gamma_{p^i}(t) = n! \cdot \tbinom{(n+1)p^i}{p^i}\gamma_{(n+1)p^i}(t),$$

since Lucas' Theorem shows that $\binom{(n+1)p^i}{p^i} \equiv n+1 \pmod p$. In particular $t_i^p = 0$ for all i.

Proposition 3.2. *Let n be a positive integer and $n = \sum_{i=0}^{s} n_i p^i$ its writing in basis p. Then:*

$$\gamma_n(t) = \gamma_{n_0}(t) \cdot \gamma_{n_1 p}(t) \cdots \gamma_{n_s p^s}(t) = \frac{t_0^{n_0}}{n_0!} \cdot \frac{t_1^{n_1}}{n_1!} \cdots \frac{t_s^{n_s}}{n_s!}.$$

Proof. The first equality is proved by induction on s using the fact that if $n = a + bp$ with $0 \leq a < p$, then $\gamma_a\gamma_{bp} = \gamma_n$, since $\binom{a+bp}{a} \equiv 1 \pmod p$. The second equality then follows from the relations $t_i^{n_i} = n_i! \cdot \gamma_{n_i p^i}(t)$. □

A corollary of the above proposition is that elements of $\ell[[t]]^{\mathrm{dp}}$ can be alternatively described as infinite sums of monomials $a_{n_0,\ldots,n_s} \cdot t_0^{n_0} \cdot t_1^{n_1} \cdots t_s^{n_s}$ where the n_i's are integers in the range $[0, p)$ and the coefficient $a_{n_0,\ldots,n_s}$ lies in ℓ. The product in $\ell[[t]]^{\mathrm{dp}}$ is then the usual product of series subject to the additional rules $t_i^p = 0$ for all i.

More precisely, restricting ourselves to some given precision of the form $N = np^s$, we deduce from the above discussion the following corollary.

Corollary 3.3. *For $N = np^s$, with $s \in \mathbb{N}$ and $n \in \{1, \ldots, p\}$, there is a canonical isomorphism of ℓ-algebras:*

$$\ell[[t]]^{\mathrm{dp}}/\ell[[t]]^{\mathrm{dp}}_{\geq N} \simeq \ell[t_0, \ldots, t_s]/(t_0^p, \ldots, t_{s-1}^p, t_s^n).$$

For instance, if we take $s = 0$ and $N = n$ in $\{1, \ldots, p\}$, we obtain the isomorphism $\ell[[t]]^{\mathrm{dp}}/\ell[[t]]^{\mathrm{dp}}_{\geq N} \simeq \ell[t]/t^N$.

In terms of complexity, the change of bases between left- and right-hand sides can both be done in $O\tilde{}(N)$ operations in ℓ: all the factorials we need can be computed once and for all for $O(\min(N, p))$ operations; then each monomial conversion takes $O(s) = O(\log(N))$ operations, for a total of $O(N \log(N)) = O\tilde{}(N)$.

The previous corollary is useful in order to devise a multiplication algorithm for divided powers, since it reduces this question to multivariate power series multiplication (addition takes linear time in both bases). To multiply in $\ell[t_0, \ldots, t_s]/(t_0^p, \ldots, t_{s-1}^p, t_s^n)$, one can use a direct algorithm: multiply and discard unwanted terms. Using for instance

Kronecker's substitution and FFT-based univariate arithmetic, we find that a multiplication in $\ell[[t]]^{\mathrm{dp}}$ at precision N (*i.e.* modulo $\ell[[t]]^{\mathrm{dp}}_{\geq N}$) can be performed with $O^{\sim}(2^{\log_p N} N)$ operations in k. A solution that leads to a cost $N^{1+\varepsilon}$ for any $\varepsilon > 0$ is in [28], but the former result will be sufficient.

3.3 The Cauchy–Lipschitz Theorem

A nice feature of the ring $\ell[[t]]^{\mathrm{dp}}$ — which does not hold for $\ell[[t]]$ notably — is the existence of an analogue of the classical Cauchy–Lipschitz theorem. This property will have a fundamental importance for the purpose of our paper; see for instance [21, Proposition 4.2] for similar considerations.

PROPOSITION 3.4. *Let $Y' = AY$ be a differential system of dimension r with coefficients in $\ell[[t]]^{\mathrm{dp}}$. For all initial data $V \in \ell^r$ (considered as a column vector) the following Cauchy problem has a unique solution in $\ell[[t]]^{\mathrm{dp}}$:*

$$\begin{cases} Y' = A \cdot Y \\ Y(0) = V. \end{cases}$$

PROOF. Let us write the expansions of A and Y:

$$A = \sum_{i=0}^{\infty} A_i \gamma_i(t) \quad \text{and} \quad Y = \sum_{i=0}^{\infty} Y_i \gamma_i(t)$$

where the A_i's and Y_i's have coefficients in ℓ. The Cauchy problem translates to $Y_0 = V$ and $Y_{n+1} = \sum_{i=0}^{n} \binom{n}{i} \cdot A_i \cdot Y_{n-i}$. It is now clear that it has a unique solution. □

Of course, Proposition 3.4 extends readily to the case where the initial data V is any matrix having r rows. In particular, taking $V = I_r$ (the identity matrix of size r), we find that there exists a unique $r \times r$ matrix Y with coefficients in $\ell[[t]]^{\mathrm{dp}}$ such that $Y(0) = I_r$ and $Y' = A \cdot Y$. This matrix is often called a *fundamental system of solutions*.

Finding solutions using Newton iteration. In characteristic zero, it is possible to compute power series solutions of a differential system such as $Y' = A \cdot Y$ using Newton iteration; an algorithm for this is presented on [5, Fig. 1].

One can use this algorithm to compute a fundamental system of solutions in our context. For this, we first need to introduce two notations. Given an element $f \in \ell[[t]]^{\mathrm{dp}}$ written as $f = \sum_i a_i \gamma_i(t)$ together with an integer m, we set $\lceil f \rceil^m = \sum_{i=0}^{m-1} a_i \gamma_i(t)$. Similarly, if M is a matrix with coefficients in $\ell[[t]]^{\mathrm{dp}}$, we define $\lceil M \rceil^m$ and $\int M$ by applying the corresponding operations entry-wise.

Algorithm `fundamental_solutions`
Input: a differential system $Y' = AY$, an integer N
Output: the fund. system of solutions modulo $\ell[[t]]^{\mathrm{dp}}_{\geq N}$

1. $Y = I_r + t\,A(0)$; $Z = I_r$; $m = 2$
2. **while** $m \leq N/2$:
3. $\quad Z = Z + \lceil Z(I_r - YZ) \rceil^m$
4. $\quad Y = Y - \Big\lceil Y\big(\int Z \cdot (Y' - \lceil A \rceil^{2m-1} Y)\big) \Big\rceil^{2m}$
5. $\quad m = 2m$
6. **return** Y

Correction is proved as in the classical case [5, Lemma 1].

Let us take $n \in \{2, \ldots, p\}$ and $s \in \mathbb{N}$ such that $n-1$ is the last digit of N written in basis p, and s the corresponding exponent; then, we have $(n-1)p^s \leq N < np^s$. Since we are only interested in costs up to logarithmic factors, we may assume that we do all operations at precision np^s (a better analysis would take into account the fact that the precision grows quadratically).

By Corollary 3.3 and the discussion that follows, arithmetic operations in $\ell[[t]]^{\mathrm{dp}}/\ell[[t]]^{\mathrm{dp}}_{\geq np^s}$ take time $O^{\sim}(2^{\log_p N} N)$. This is also the case for differentiation and integration, in view of the formulas given in the previous subsection; truncation is free. The total complexity of Algorithm `fundamental_solutions` is therefore $O^{\sim}(2^{\log_p N} N r^{\omega})$ operations in ℓ, where r is the dimension of the differential system. If $N = p^{O(1)}$, which is what we need later on, this is $O^{\sim}(Nr^{\omega})$.

The case of differential operators. We now consider the case of the differential system associated to a differential operator $L = a_r \partial^r + \cdots + a_1 \partial + a_0 \in \ell[[t]]^{\mathrm{dp}}\langle \partial \rangle$. We will work under the following few assumptions: we assume that a_r is invertible, and that there exists an integer $d < p$ such that all a_i's can be written $a_i = \alpha_{i,0} + \alpha_{i,1}\gamma_1(t) + \cdots + \alpha_{i,d}\gamma_d(t)$ for some coefficients $\alpha_{i,j}$ in ℓ; thus, by assumption, $\alpha_{r,0}$ is a unit in ℓ. Our goal is still to compute a basis of solutions up to precision N; the algorithm is a direct adaptation of a classical construction to the case of divided powers.

In all that follows, we let $f_0, \ldots, f_{r-1}$ be the solutions of L in $\ell[[t]]^{\mathrm{dp}}$, such that f_i is the unique solution of the Cauchy problem (*cf* Proposition 3.4):

$$L(f_i) = 0 \quad ; \quad f_i^{(j)}(0) = \delta_{ij} \quad \text{for } 0 \leq j < r \qquad (7)$$

where δ_{ij} is the Kronecker delta. For $f = \sum_{j=0}^{\infty} \xi_j \gamma_j(t)$ in $\ell[[t]]^{\mathrm{dp}}$, a direct computation shows that the n-th coefficient of $L(f)$ is $\sum_{i=0}^{r} \sum_{j=0}^{n} \binom{n}{j} \alpha_{i,j} \xi_{n+i-j}$. Assume $L(f) = 0$. Then, extracting the term in ξ_{n+r}, and using that $\alpha_{i,j} = 0$ for $j > d$, we get $\xi_{n+r} = \frac{-1}{\alpha_{r,0}} \sum_{i=0}^{r-1} \sum_{j=0}^{d} \binom{n}{j} \alpha_{i,j} \xi_{n+i-j}$. Letting $m = i - j$, we find $\xi_{n+r} = \sum_{m=-d}^{r-1} A_m(n) \xi_{n+m}$ with

$$A_m(n) = \frac{-1}{\alpha_{r,0}} \sum_{i=0}^{r-1} \binom{n}{i-m} \alpha_{i,i-m} = \sum_{\substack{0 \leq i \leq r-1 \\ 0 \leq i-m \leq d}} \frac{-\alpha_{i,i-m}}{\alpha_{r,0}(i-m)!}\, n^{\underline{i-m}}$$

and $n^{\underline{i-m}} = n(n-1)\cdots(n-(i-m-1))$ is a falling factorial. The expression above for A_m is well-defined, since we assumed that $d < p$, and shows that A_m is a polynomial of degree at most d.

From this, writing the algorithm is easy. We need two subroutines: `from_falling_factorial`(F), which computes the expansion on the monomial basis of a polynomial of the form $F = \sum_{0 \leq j \leq n} f_j n^{\underline{j}}$, and `eval`$(F, N)$, which computes the values of a polynomial F at the N points $\{0, \ldots, N-1\}$. The former can be done using the divide-and-conquer algorithm of [8, Section 3] in time $O^{\sim}(n)$; the latter by the algorithm of [17, Chapter 10], in time $O^{\sim}(\deg(F) + N)$. The previous discussion leads to the algorithm `solutions_operator` below. In view of the previous discussion, the cost analysis is straightforward (at step 2., notice that all required factorials can be computed in time $O(d)$). The costs reported in the pseudo-code indicate the *total* amount of time spent at the corresponding line.

Algorithm `solutions_operator`
Input: a differential operator $L \in \ell[[t]]^{\mathrm{dp}}\langle \partial \rangle$ of bidegree (d, r), with $d < p$; an integer N
Output: the solutions $f_0, \ldots, f_{r-1}$ at precision N

1. **for** $m = -d, \dots, r-1$:
2. $\quad \hat{A}_m = \sum_{0 \le i \le r-1, 0 \le i-m \le d} \frac{-\alpha_{i,i-m}}{\alpha_{r,0}(i-m)!}\, x^{\underline{i-m}}$
   $\quad$ COST: $O(d(r+d))$
3. $\quad A_m = \texttt{from\_falling\_factorial}(\hat{A}_m)$
   $\quad$ COST: $O^{\sim}(d(r+d))$
4. $\quad$ Store $\texttt{eval}(A_m, N-r)$
   $\quad$ COST: $O^{\sim}((d+N)(r+d))$
5. **for** $i = 0, \dots, r-1$:
6. $\quad f_i = [0, \dots, 0, 1, 0, \dots, 0]$ (ith unit vector of length r)
   $\quad$ COST: $O(r^2)$
7. $\quad$ **for** $n = 0, \dots, N-r-1$:
8. $\quad\quad f_{i,n+r} = \sum_{m=-d}^{r-1} A_m(n) f_{i,n+m}$
   $\quad\quad$ COST: $O(rN(r+d))$
9. **return** $f_0, \dots, f_{r-1}$

Altogether, we obtain the following result, where we use the assumption $N > d$ to simplify slightly the cost estimate.

LEMMA 3.5. *Suppose that $p < d$. Given a positive $N > d$, the classes of $f_0, \dots, f_{r-1}$ modulo $\ell[[t]]^{\mathrm{dp}}_{\ge N}$ can be computed with at most $O(rN(r+d))$ operations in ℓ.*

In particular, Algorithm `solutions_operator` has a better cost than `fundamental_solutions` when $d = O(r^{\omega-1})$.

4. COMPUTING THE P-CURVATURE

In all this section, we work over a field k of characteristic $p > 0$. We consider a differential system $Y' = AY$ of dimension r and denote by A_p the matrix of its p-curvature. We write $A = \frac{1}{f_A}\tilde{A}$, where f_A is in $k[x]$ and $\tilde{A}$ is a matrix with polynomial entries. Let $d = \max(\deg f_A, \deg \tilde{A})$, where $\deg \tilde{A}$ is the maximal degree of the entries of $\tilde{A}$. We recall ([13, Prop. 3.2], [9, Lemma 1]) a bound on the size of A_p. The bound follows from the recurrence (1), and it is tight.

LEMMA 4.1. *The entries of the matrix $f_A^p \cdot A_p$ are all polynomials of degree at most dp.*

The goal of this section is to prove the following theorem.

THEOREM 4.2. *There exists an algorithm (presented below) which computes the matrix of the p-curvature of the differential system $Y' = AY$ in $O^{\sim}(pdr^{\omega})$ operations in k.*

It is instructive to compare this cost with the size of the output. By Lemma 4.1, the latter is an $r \times r$ matrix whose entries are rational functions whose numerator and denominator have degree $\simeq pd$, so its size is roughly pdr^2 elements of k. Our result $O^{\sim}(pdr^{\omega})$ is quasi-optimal if we assume that matrix multiplication can be performed in quasi-optimal time.

4.1 A formula for the p-curvature

Let A_p denote the matrix of the p-curvature of the differential system $Y' = AY$ (in the usual monomial basis). The expression of A_p given at the very end of §2 is unfortunately not well-suited for fast computation. The aim of this subsection is to give an alternative formula for A_p using the framework of divided powers.

In order to relate $k(x)$ and a ring $\ell[[t]]^{\mathrm{dp}}$, we pick a separable polynomial $S \in k[x]$ which is coprime with f_A and set $\ell = k[x]/S$ (which is thus not necessarily a field). Let $a \in \ell$ be the class of x. We consider the ring homomorphism:

$$\begin{array}{rccc} \varphi_S : & k[x] & \to & \ell[t]/t^p \\ & f(x) & \mapsto & f(t+a) \bmod t^p. \end{array}$$

Regarding the differential structure, we observe that φ_S commutes with the derivation when $\ell[t]/t^p$ is endowed with the standard derivation $\frac{d}{dt}$. We furthermore deduce from the fact that S and f_A are coprime that φ_S extends to a homomorphism of differential rings $k[x][\frac{1}{f_A}] \to \ell[t]/t^p$ that we continue to denote by φ_S. We set $\psi_S = \iota \circ \varphi_S$ where ι is the canonical inclusion $\ell[t]/t^p \hookrightarrow \ell[[t]]^{\mathrm{dp}}$ (*cf* §3). As before, ψ_S commutes with the derivation. Finally, because S is separable, we can check that φ_S is surjective and its kernel is the ideal generated by S^p. Hence φ_S induces an isomorphism:

$$k[x]/S^p = k[x][\tfrac{1}{f_A}]/S^p \xrightarrow{\sim} \ell[t]/t^p. \tag{8}$$

Let Y_S be a fundamental system of solutions of the differential system $Y' = \psi_S(A) \cdot Y$, *i.e.* Y_S is an $r \times r$ matrix with coefficients in $\ell[[t]]^{\mathrm{dp}}$ such that $Y_S(0) = I_r$ and $Y_S' = \psi_S(A) \cdot Y_S$. The existence of Y_S is guaranteed by Proposition 3.4. Moreover, the matrix Y_S is invertible because $Y_S(0) = I_r$ is.

PROPOSITION 4.3. *Keeping the above notations, we have:*

$$\varphi_S(A_p) = -Y_S^{(p)} \cdot Y_S^{-1} \tag{9}$$

where $Y_S^{(p)}$ is the matrix obtained from Y_S by taking the p-th derivative entry-wise.

PROOF. We set $Z_S = Y_S^{-1}$ and let (M, ∂) denote the differential module over $\ell[[t]]^{\mathrm{dp}}$ associated to the differential system $Y' = \psi_S(A)Y$. Let $y_1, \dots, y_r$ denote the column vectors of Y_S. They are all solutions of the system $Y' = \psi_S(A)Y$, meaning that $\partial(y_i) = 0$ for all i. Furthermore, if $(e_1, \dots, e_r)$ is the canonical basis of $(\ell[[t]]^{\mathrm{dp}})^r$, we have the matrix relations: ${}^{\mathrm{t}}Y_S \cdot \underline{e} = \underline{y}$ and $\underline{e} = {}^{\mathrm{t}}Z_S \cdot \underline{y}$ where $\underline{y}$ (resp. $\underline{e}$) is the column vector whose coordinates are the *vectors* y_i's (resp. the e_i's). Applying ∂ to the above relation, we find $\partial(\underline{e}) = {}^{\mathrm{t}}Z_S' \cdot \underline{y} + {}^{\mathrm{t}}Z_S \cdot \partial(\underline{y}) = {}^{\mathrm{t}}Z_S' \cdot \underline{y}$ and iterating this p times, we deduce $\partial^p(\underline{e}) = {}^{\mathrm{t}}Z_S^{(p)} \cdot \underline{y} = {}^{\mathrm{t}}Z_S^{(p)} \cdot {}^{\mathrm{t}}Y_S \cdot \underline{e}$. On the other hand, the matrix $\psi_S(A_p)$ of the p-curvature is defined by the relation $\partial^p(\underline{e}) = {}^{\mathrm{t}}\psi_S(A_p) \cdot \underline{e}$. Therefore we get $\psi_S(A_p) = Y_S \cdot Z_S^{(p)}$. Now differentiating p times the relation $Y_S Z_S = I_r$, we find $Y_S^{(p)} Z_S + Y_S \cdot Z_S^{(p)} = 0$. Combining this with the above formula for $\psi_S(A_p)$ concludes the proof. □

In our setting, the matrix A_p has coefficients in $k[x][\frac{1}{f_A}]$ (*cf* Lemma 4.1), from which we deduce that $\psi_S(A_p)$ has actually coefficients in the subring $\ell[t]/t^p$ of $\ell[[t]]^{\mathrm{dp}}$. Therefore, using Eq. (9), one can compute $\psi_S(A_p)$ knowing only Y_S modulo the ideal $\ell[[t]]^{\mathrm{dp}}_{\ge 2p}$.

One can actually go further in this direction and establish a variant of Eq. (9) giving an expression of $\psi_S(A_p)$ which involves only the reduction of Y_S modulo $\ell[[t]]^{\mathrm{dp}}_{\ge p}$. To make this precise, we need an extra notation. Given an integer $i \in [0, p)$ and a polynomial $f \in \ell[t]/t^p$ (resp. a matrix M with coefficients in $\ell[t]/t^p$), we write $\mathrm{Coeff}(f, i)$ (resp. $\mathrm{Coeff}(M, i)$) for the coefficient in t^i in f (resp. in M).

PROPOSITION 4.4. *Keeping the above notations, we have:*

$$\begin{aligned}\psi_S(A_p) &= -\bar{Y}_S \cdot Y_S^{(p)}(0) \cdot \bar{Y}_S^{-1} \\ &= \bar{Y}_S \cdot \mathrm{Coeff}(A \cdot \bar{Y}_S,\, p{-}1) \cdot \bar{Y}_S^{-1} \qquad (10)\end{aligned}$$

where we have set $\bar{Y}_S = Y_S \mod \ell[[t]]^{\mathrm{dp}}_{\geq p}$.

PROOF. Differentiating p times the relation $Y_S' = \psi_S(A) \cdot Y_S$, we observe that $Y_S^{(p)}$ is solution of the same differential system $Y' = \psi_S(A)Y$. Hence, thanks to uniqueness in Cauchy–Lipschitz Theorem, we have the relation $Y_S^{(p)} = Y_S \cdot Y_S^{(p)}(0)$. The first part of the Proposition follows by plugging this in Eq. (9) and reducing the result modulo $\ell[[t]]^{\mathrm{dp}}_{\geq p}$. To establish the second part, it is now enough to notice that the relation $Y_S' = \psi_S(A) \cdot Y_S$ implies:

$$Y_S^{(p)}(0) = (A \cdot Y_S)^{(p-1)}(0) = -\mathrm{Coeff}(A \cdot \bar{Y}_S,\, p{-}1)$$

the minus sign coming from $(p-1)! \equiv -1 \pmod p$. □

REMARK 4.5. *We can rephrase Proposition 4.4 as follows: letting* $y_1, \ldots, y_r$ *denote the column vectors of* Y_S *and* $\bar{y}_i \in (\ell[t]/t^p)^r$ *be the reduction of* y_i*, the p-curvature of A modulo* t^p *is the linear endomorphism of* $(\ell[t]/t^p)^r$ *whose matrix in the basis* $(\bar{y}_1, \ldots, \bar{y}_r)$ *is* $\mathrm{Coeff}(A \cdot \bar{Y}_S,\, p{-}1)$*. It is worth remarking that the latter matrix has coefficients in the subring* ℓ *of* $\ell[t]/t^p$*.*

Remembering Eq. (8), we conclude that Proposition 4.4 allows us to compute the image of the p-curvature A_p modulo S^p. The strategy of our algorithm now becomes clear: we first compute A_p modulo S^p for various polynomials S and, when we have collected enough congruences, we put them together to reconstruct A_p. The first step is detailed in §4.2 just below and the second step is the subject of §4.3.

4.2 Local calculations

In all this subsection, we fix a separable polynomial $S \in k[x]$ and denote by m its degree. Our goal is to design an algorithm for computing the matrix A_p modulo S^p. After Proposition 4.4, the main remaining algorithmic issue is the effective computation of the isomorphism φ_S and its inverse.

Applying φ_S and its inverse. We remark that φ_S factors as follows:

$$\begin{array}{ccccc} k[x]/S^p & \to & k[x,t]/\langle S, (t-x)^p\rangle & \to & k[x,t]/\langle S, t^p\rangle \\ x & \mapsto & t & \mapsto & t+a. \end{array}$$

Applying the right-hand mapping, or its inverse, amounts to doing a polynomial shift in degree p with coefficients in $k[x]/S$. Using the divide-and-conquer algorithm of [16], this can be done in $O\tilde{}(p)$ arithmetic operations in $k[x]/S$, which is $O\tilde{}(pm)$ operations in k. Thus, we are left with the left-hand factor, say $\varphi_S^\star$. Applying it is straightforward and can be achieved in $O\tilde{}(pm)$ operations in k. It then only remains to explain how one can apply efficiently ${\varphi_S^\star}^{-1}$.

We start by determining the image of x by ${\varphi_S^\star}^{-1}$; call it $y = {\varphi_S^\star}^{-1}(x)$; we may identify it with its canonical preimage in $k[x]$, which has degree less than pm. Write $y = \sum_{0\leq i<p} \zeta_i(x^p)x^i$, with every ζ_i in $k[x]$ of degree less than m (so that $\zeta_i(x^p)$ has degree less than pm). Its image through $\varphi_S^\star$ is $\sum_{0\leq i<p} \zeta_i(t^p)t^i$, which is $\sum_{0\leq i<p} \zeta_i(x^p)t^i$, since $x^p = t^p$ in $k[x,t]/\langle S, (t-x)^p\rangle$.

Since $\varphi_S^\star(y) = x$, we deduce that $\zeta_0(x^p) = x \bmod S$ and $\zeta_i(x^p) = 0 \bmod S$ for $i = 1, \ldots, p-1$. The first equality implies that x^p generates $k[x]/S$, so the fact that ζ_0 has degree less than m implies that ζ_0 is the unique polynomial with this degree constraint such that $\zeta_0(x^p) = x \bmod S$. The other equalities then imply that $\zeta_i = 0$ for $i = 1, \ldots, p-1$.

In order to compute ζ_0, we first compute $\nu = x^p \bmod S$, using $O\tilde{}(m\log(p))$ operations in k. Then, we have to find the unique polynomial ζ_0 of degree less than m such that $\zeta_0(\nu) = x \bmod S$. In general, one can compute ζ_0 in $O(m^\omega)$ operations in k by solving a linear system. In the common case where $m < p$, there exists a better solution. Indeed, denote by $\mathrm{tr} : k[x]/S \to k$ the k-linear trace form and write $t_i = \mathrm{tr}(\nu^i)$ and $t_i' = \mathrm{tr}(x\nu^i)$, for $i = 0, \ldots, m-1$. Then formulas such as those in [25] allow us to recover ζ_0 from $\mathbf{t} = (t_0, \ldots, t_{m-1})$ and $\mathbf{t}' = (t_0', \ldots, t_{m-1}')$ in time $O\tilde{}(m)$. These formulas require that $m < p$ and that S' be invertible modulo S, which is ensured by our assumption that S is separable. To compute $\mathbf{t}$ and $\mathbf{t}'$, we can use Shoup's power projection algorithm [29], which takes $O(m^{(\omega+1)/2})$ operations in k.

Once ζ_0 is known, to apply the mapping ${\varphi_S^\star}^{-1}$ to an element $g(x,t)$, we proceed coefficient-wise in t. Write $g = \sum_{0\leq i<p} g_i(x)t^i$, with all g_i of degree less than m. Then ${\varphi_S^\star}^{-1}(g) = \sum_{0\leq i<p} (g_i(\zeta_0) \bmod T)\,(x^p)\,x^i$ where T is the polynomial obtained by raising all coefficients of S to the power p, so that $S(x)^p = T(x^p)$.

Computing T takes $O(m\log(p))$ operations in k; then, computing each term $g_i(\zeta_0) \bmod T$ can be done using the Brent-Kung modular composition algorithm for $O(m^{(\omega+1)/2})$ operations in k; the total is $O(m^{(\omega+1)/2}p)$. Finally, the evaluation at x^p and the summation needed to obtain ${\varphi_S^\star}^{-1}(g)$ do not involve any arithmetic operations.

REMARK 4.6. *In the case where* $S = x^m - c$ *(where* $c \in k$ *and* p *does not divide* m*), there actually exists a quite simple explicit formula for* ${\varphi_S^\star}^{-1}$*: it takes* t *to* x *and* x *to* $c^q x^{pn}$ *where* n *and* q *are integers satisfying the Bézout's relation* $pn + qm = 1$*. Using this, one can compute* ${\varphi_S^\star}^{-1}(g)$ *in* $O\tilde{}(pm)$ *operations in* k *in this special case.*

Conclusion. Let us call `phiS` and `phiS_inverse` the two subroutines described above for computing φ_S and its inverse respectively. Proposition 4.4 leads to the following algorithm for computing the p-curvature modulo S^p.

Algorithm `local_p_curvature`
Input: a polynomial S and a matrix $A_S \in M_r(k[x]/S^p)$
Output: the p-curvature of the system $Y' = A_S\, Y$

1. $A_{S,\ell} = \mathtt{phiS}(A_S)$
   COST: $O\tilde{}(pr^2m)$ operations in k (with $m = \deg S$)
2. compute a fund. system of solutions $Y_S \in M_r(\ell[t]/t^p)$ of the system $Y' = A_{S,\ell}Y$ at precision p.
   COST: $O\tilde{}(pr^\omega)$ op. in ℓ using `fundamental_solutions`
   REMARK: Here $\ell = k[x]/S$
3. $A_{p,\ell} = Y_S \cdot \mathrm{Coeff}(AY_S,\, p{-}1) \cdot Y_S^{-1}$ at precision $O(t^p)$
   COST: $O\tilde{}(pr^\omega)$ operations in ℓ
4. $A_p = \mathtt{phiS\_inverse}(A_{p,\ell})$
   COST: $O\tilde{}(pr^2m^\omega)$ operations in k in general
   $O\tilde{}(pr^2m^{(\omega+1)/2})$ operations in k if $m < p$
5. **return** A_p.

To conclude with, it is worth remarking that implementing the algorithm `local_p_curvature` can be done using usual power series arithmetic: indeed, we only need to perform computations in the quotient $\ell[[t]]^{\mathrm{dp}}/\ell[[t]]^{\mathrm{dp}}_{\geq p}$ which is isomorphic to $\ell[t]/t^p$ by Corollary 3.3. Furthermore, we note that if we are using the algorithm `fundamental_solutions` at line 2, then Y_S^{-1} can be computed by performing an extra loop in `fundamental_solutions`; indeed the matrix Z we obtain this way is exactly Y_S^{-1}.

4.3 Gluing

We recall that we have started with a differential system $Y' = AY$ (with $A = \frac{1}{f_A}\tilde{A}$) and that our goal is to compute the matrix A_p of its p-curvature. Lemma 4.1 gives bounds on the size of the entries of A_p. We need another lemma, which ensures that we can find enough small "evaluation points" (lying in a finite extension of k). Let $\mathbb{F}_p$ denote the prime subfield of k.

LEMMA 4.7. *Given a positive integer D and a nonzero polynomial $f \in k[x]$, there exist pairwise coprime polynomials $S_1, \ldots, S_n \in \mathbb{F}_p[x]$ with $n \leq D$ such that:*
- $\sum_{i=1}^n \deg S_i \geq D$
- *for all i, the polynomial S_i is coprime with f and has degree at most $1 + \log_p(D + \deg f)$.*

PROOF. Let m be the smallest integer such that $p^m \geq D + \deg f$. Clearly $m \leq 1 + \log_q(D + \deg f) \leq 1 + \log_p(D + \deg f)$. Let $\mathbb{F}_{p^m}$ be an extension of $\mathbb{F}_p$ of degree m and K be the compositum of k and $\mathbb{F}_{p^m}$. Let $S_1, \ldots, S_t$ be the minimal polynomials over $\mathbb{F}_p$ (without repetition) of all elements in $\mathbb{F}_{p^m} \subset K$ which are not a root of f. We then have $\deg S_i \leq m$ for all i and $\sum_{i=1}^t \deg S_i \geq p^m - \deg f \geq D$. It remains now to define n as the smallest integer such that $\sum_{i=1}^n \deg S_i \geq D$. Minimality implies $\sum_{i=1}^{n-1} \deg S_i < D$ and thus $n \leq D$. Therefore $S_1, \ldots, S_n$ satisfy all the requirements of the lemma. □

The above proof yields a concrete algorithm for producing a sequence $S_1, \ldots, S_n$ satisfying the properties of Lemma 4.7: we run over elements in $\mathbb{F}_{p^m}$ and, for each new element, append its minimal polynomial over $\mathbb{F}_p$ to the sequence (S_i) unless it is not coprime with f. We continue this process until the condition $\sum_{i=1}^n \deg S_i \geq D$ holds. Keeping in mind the logarithmic bound on m, we find that the complexity of this algorithm is at most $O^{\tilde{}}(D + \deg f)$ operations in k. Let us call `generate_points` the resulting routine: it takes as input the parameters f and D and return an admissible sequence $S_1, \ldots, S_n$.

We are now ready to present our algorithm for computing the p-curvature:

Algorithm `p_curvature`
Input: a matrix A written as $A = \frac{1}{f_A} \cdot \tilde{A}$
Output: the p-curvature of the differential system $Y' = AY$

1. $S_1, \ldots, S_n =$ `generate_points`$(f_A, d+1)$
   COST: $O^{\tilde{}}(d)$ operations in k
   REMARK: we have $n = O(d)$ and $\deg S_i = O(\log d)$, $\forall i$
2. **for** $i = 1, \ldots, n$:
   $A_{i,p} =$ `local_p_curvature`$(S_i, A \bmod S_i^p)$
   COST: $O^{\tilde{}}(pdr^\omega)$ operations in k
3. compute $B \in M_r(k[x])$ with entries of degree $\leq pd$ such that $B \equiv f_A^p \cdot B_i \pmod{S_i^p}$ for all i
   COST: $O^{\tilde{}}(pdr^2)$ operations in k
4. **return** $\frac{1}{f_A^p} \cdot B$

In view of the previous discussion and Lemma 4.1, the correctness and the cost analysis of the algorithm `p_curvature` are both straightforward. Hence, Theorem 4.2 is proved.

We conclude this subsection with three remarks. First, when applying Chinese Remainder Theorem (CRT) on line 3 of Algorithm `p_curvature`, we notice that all moduli S_i^p are polynomials in x^p. This allows the following optimization. Writing $f_A^p \cdot B_i \equiv \sum_{j=0}^{p-1} B_{i,j}(x^p)x^j \pmod{S_i^p(x)}$ and denoting by C_j the unique solution of degree at most d to the congruence system:

$$B_j(x) \equiv B_{i,j}(x) \pmod{T_i(x)} \quad \text{where } T_i(x^p) = S_i^p(x)$$

we have $B = \sum_{j=0}^{p-1} B_j(x)x^j$. This basically allows us to replace one CRT with polynomials of degree dp by p CRT with polynomials of degree d. We save this way the polynomial factors in $\log(p)$ in the complexity.

Second, instead of working with n polynomials S_i, one may alternatively choose a unique polynomial S of the form $S = X^m - a$ where $m \geq d$ is an integer not divisible by p and $a \in k$ are such that S and f_A are coprime. This avoids the use of Chinese Remainder Theorem and the resulting complexity stays in $O^{\tilde{}}(pdr^\omega)$ provided that we use Remark 4.6 in order to compute the inverse of φ_S.

Third, we observe that the algorithm `p_curvature` is very easily parallelizable. Indeed, each iteration of the main loop (on line 2) is completely independent from the others. Thus, they all can be performed in parallel. Moreover, according to the first remark (just above), the application of the Chinese Remainder Theorem (on line 3) splits into pr^2 smaller independent problems and can therefore be efficiently parallelized as well.

4.4 The case of differential operators

To conclude with, we would like to discuss the case of a differential operator $L = a_r\partial^r + a_{r-1}\partial^{r-1} + \cdots + a_1\partial + a_0$ with $a_i \in k[x]$ for all i, of maximal degree d.

Recall that the p-curvature of L is that of the differential module $(\mathfrak{A}\langle\partial\rangle/\mathfrak{A}\langle\partial\rangle L, \partial_{-C})$, where C is the companion matrix associated to L as in (3). Applying directly the formulas in Proposition 4.4 requires the knowledge of the solutions of the system $Y' = -CY$. It is in fact easier to compute solutions for the system $X' = {}^tCX$, since we saw that these solutions are the vectors of the form ${}^t(y, y', \ldots, y^{(r-1)})$, where y is a solution of L. This is however harmless: the p-curvatures A_p and B_p of the respective systems $Y' = -CY$ and $X' = {}^tCX$ (which are so-called adjoint) satisfy $A_p = -{}^tB_p$. Thus, we can use the formulas given above to compute $\varphi_S(B_p)$, and deduce $\varphi_S(A_p)$ for a negligible cost. Equivalently, one may notice that the fundamental matrices of solutions of our two systems are transpose of one another, up to sign.

Moreover, instead of using the second formula of Proposition 4.4 to compute the local p-curvatures, we recommend using the first one, which is $\varphi_S(B_p) = -X_S \cdot X_S^{(p)}(0) \cdot X_S^{-1}$ where X_S is a fundamental system of solutions of $X' = {}^tCX$ and $\bar{X}_S$ denotes its reduction in $M_r(\ell[t]/t^p)$. If $f_0, \ldots, f_{r-1}$ are solutions of the system (7), the (i,j)-th entry of X_S is just $f_j^{(i)}$. Hence the matrices $\bar{X}_S$ and $X_S^{(p)}(0)$ can be obtained from the knowledge of the image of f_i's modulo

		p							
		157	**281**	**521**	**983**	**1 811**	**3 433**	**6 421**	**12 007**
$d = 5,$	$r = 5$	0.39 s	0.71 s	1.22 s	2.34 s	4.41 s	8.93 s	18.0 s	36.1 s
		0.26 s	0.76 s	2.69 s	9.05 s	32.6 s	145 s	593 s	2 132 s
$d = 5,$	$r = 11$	1.09 s	2.05 s	3.65 s	7.05 s	12.6 s	26.7 s	53.3 s	109 s
		1.25 s	3.70 s	12.8 s	45.5 s	163 s	725 s	2 942 s	–
$d = 5,$	$r = 20$	2.93 s	5.25 s	9.52 s	17.7 s	32.5 s	68.1 s	139 s	288 s
		4.29 s	12.4 s	42.5 s	153 s	548 s	2 460 s	–	–
$d = 11,$	$r = 20$	6.89 s	13.3 s	22.6 s	45.0 s	80.4 s	167 s	342 s	711 s
		11.6 s	34.7 s	121 s	486 s	1 943 s	–	–	–
$d = 20,$	$r = 20$	14.0 s	25.1 s	49.9 s	94.0 s	176 s	357 s	733 s	1 472 s
		27.0 s	84.5 s	314 s	1 283 s	–	–	–	–

Running times obtained with Magma V2.19-4 on an AMD Opteron 6272 machine at 2GHz and 8GB RAM, running Linux.

Figure 1: Average running time on random inputs of various sizes

$\ell[[t]]^{\mathrm{dp}}_{\geq p+r}$ just by reorganizing coefficients (and possibly multiplying by some factorials depending on the representation of elements of $\ell[[t]]^{\mathrm{dp}}$ we are using).

As for the f_i's, they can be computed by the algorithm `solutions_operator` (provided its assumptions are satisfied). We need finally to compute X_S^{-1}: since $X_S(0)$ is the identity matrix, this can be done either using Newton iterator, a divide-and-conquer approach or a combination of both, which computes the inverse of X_S at a small precision, and uses divide-and-conquer techniques for higher ones (the latter being the most efficient in practice). All these remarks do speed up the execution of our algorithms when d is not too large compared to r.

Last but not least, we notice that, in the case of differential operators, the matrix A_p is easily deduced from its first column. Indeed, writing $A_p = (a_{i,j})_{0\leq i,j<r}$ and letting $c_j = a_{r-1,j}\partial^{r-1} + \cdots + a_{1,j}\partial + a_{0,j} \in k(x)\langle\partial\rangle$ be the differential operator obtained from the j-column of A_p, it is easily checked that c_{j+1} is the remainder in the Euclidean division of ∂c_j by L. Comparing orders, we further find $c_{j+1} = \partial c_j - \frac{\mathrm{lc}(c_j)}{a_r} L$ where $\mathrm{lc}(c_j)$ is the leading coefficient of c_j. This remark is interesting because it permits to save memory: indeed, instead of storing all local p-curvatures $A_{p,\ell}$, we can just store their first column. Doing this, we can reconstruct the first column of A_p using the Chinese Remainder Theorem (*cf* §4.3) and then compute the whole matrix A_p using the recurrence.

5. IMPLEMENTATION AND TIMINGS

We implemented our algorithms in Magma in the case of differential operators; the source code is available at https://github.com/schost/. Figure 1 gives running times for random operators of degrees (d, r) in $k[x]\langle\partial\rangle$ and compares them with running times of (a fraction free version of) Katz's algorithm which consists in computing the recursive sequence (A_i) until $i = p$. In each cell, the first line (resp. the second line) corresponds to the running time obtained with our algorithm (resp. Katz's algorithm); a dash indicates that the corresponding running time exceeded one hour. Our benchmarks rather well reflect the predicted dependence with respect to p: quasi-linear for our algorithm and quadratic for Katz's algorithm.

Larger examples (than those presented in Fig. 1) are also reachable: for instance, we computed the first column of the p-curvature of a "small" multiple of the operator $\phi_H^{(5)}$ considered in [10, Appendix B.3] modulo the prime 27449. This operator has bidegree $(d, r) = (108, 28)$. The computation took about 19 hours and the size of the output in human-readable format is about 1GB (after bzip2 compression, it decreases to about 300MB).

6. REFERENCES

[1] P. Berthelot. *Cohomologie cristalline des schémas de caractéristique $p > 0$.* Lecture Notes in Mathematics, Vol. 407. Springer-Verlag, Berlin-New York, 1974.

[2] P. Berthelot and A. Ogus. *Notes on crystalline cohomology.* Princeton University Press, Princeton, N.J.; University of Tokyo Press, Tokyo, 1978.

[3] A. Bostan, S. Boukraa, S. Hassani, J.-M. Maillard, J.-A. Weil, and N. Zenine. Globally nilpotent differential operators and the square Ising model. *J. Phys. A*, 42(12):125206, 50, 2009.

[4] A. Bostan, X. Caruso, and E. Schost. A fast algorithm for computing the characteristic polynomial of the p-curvature. In *ISSAC'14*, pages 59–66. ACM, New York, 2014.

[5] A. Bostan, F. Chyzak, F. Ollivier, B. Salvy, É. Schost, and A. Sedoglavic. Fast computation of power series solutions of systems of differential equations. In *18th ACM-SIAM Symposium on Discrete Algorithms*, pages 1012–1021, 2007. New Orleans, January 2007.

[6] A. Bostan and M. Kauers. Automatic classification of restricted lattice walks. In *FPSAC'09*, DMTCS Proc., AK, pages 201–215. 2009.

[7] A. Bostan and M. Kauers. The complete generating function for Gessel walks is algebraic. *Proc. Amer. Math. Soc.*, 138(9):3063–3078, 2010. With an appendix by Mark van Hoeij.

[8] A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points. *J. Complexity*, 21(4):420–446, 2005.

[9] A. Bostan and É. Schost. Fast algorithms for differential equations in positive characteristic. In *ISSAC'09*, pages 47–54. ACM, New York, 2009.

[10] S. Boukraa, S. Hassani, J.-M. Maillard, and N. Zenine. Singularities of n-fold integrals of the Ising class and the theory of elliptic curves. *J. Phys. A*, 40(39):11713–11748, 2007.

[11] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28(7):693–701, 1991.

[12] A. Chambert-Loir. Théorèmes d'algébricité en géométrie diophantienne (d'après J.-B. Bost, Y. André, D. & G. Chudnovsky). *Séminaire Bourbaki*, 282(886):175–209, 2002.

[13] T. Cluzeau. Factorization of differential systems in characteristic p. In *ISSAC'03*, pages 58–65. ACM Press, 2003.

[14] T. Cluzeau and M. van Hoeij. A modular algorithm for computing the exponential solutions of a linear differential operator. *J. Symbolic Comput.*, 38(3):1043–1076, 2004.

[15] F. L. Gall. Powers of tensors and fast matrix multiplication. In *ISSAC'14*, pages 296–303, 2014.

[16] J. von zur Gathen and J. Gerhard. Fast algorithms for Taylor shifts and certain difference equations. In *ISSAC'97*, pages 40–47. ACM, 1997.

[17] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, Cambridge, second edition, 2003.

[18] D. Harvey, J. v[redacted]r Hoeven, and G. Lecerf. Faster polynomial multiplication over finite fields. http://arxiv.org/abs/1407.3361, 2014.

[19] N. M. Katz. Algebraic solutions of differential equations (p-curvature and the Hodge filtration). *Invent. Math.*, 18:1–118, 1972.

[20] N. M. Katz. A conjecture in the arithmetic theory of differential equations. *Bull. Soc. Math. France*, (110):203–239, 1982.

[21] W. F. Keigher and F. L. Pritchard. Hurwitz series as formal functions. *J. Pure Appl. Algebra*, 146(3):291–304, 2000.

[22] M. van der Put. Differential equations in characteristic p. *Compositio Mathematica*, 97:227–251, 1995.

[23] M. van der Put. Reduction modulo p of differential equations. *Indag. Mathem.*, 7(3):367–387, 1996.

[24] M. van der Put and M. Singer. *Galois theory of linear differential equations.* Springer, 2003.

[25] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Appl. Algebra Engrg. Comm. Comput.*, 9(5):433–461, 1999.

[26] A. Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Informatica*, 7:395–398, 1977.

[27] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[28] É. Schost. Multivariate power series multiplication. In *ISSAC'05*, pages 293–300. ACM, 2005.

[29] V. Shoup. Fast construction of irreducible polynomials over finite fields. *Journal of Symbolic Computation*, 17(5):371–391, 1994.

[30] Y. Tang. Algebraic solutions of differential equations over the projective line minus three points. http://arxiv.org/abs/1412.7875, 2014.